Project 3: Implement a planning search
Research report

Batian Niebel

May 4, 2017

Implementation The loading & unloading actions, level-sum heuristics, and problem 1-3 were implemented in `my_air_cargo_proble`. The methods required to build the planning graph algorithm were implemented in `my_planning_graph.py`. The implementation was tested using the provided unit tests.

The different algorithms together with problem 1-3 were compared using `run_analysis.py`. The results were stored in `search_result.json` and further processed with the ipython notebook `analyze_results.ipynb`.

Analysis of the results All searches except the breadth first search yielded the same path length for the individual problems (cf. table 1). The optimal plans for the problems (cf listing 1, 2 & 3) was taken from the A* search with the planning graph and level-sum heuristics.

The planning graph implementation took exceptionally long, probably due to a not optimized implementation and the small size of the problems. The overhead of the planning graph does not out weight the uniformed searches. However the planning graph shined as expected in the number of expanded nodes, thus if node expansion gets more expensive (larger problems) it is for sure a feasible option.

In general the A* search with the level sum heuristics performed best. The depth first search found not the optimal solutions, due to its tendency to get stuck in local minimum, when not further expanding certain trees, it is a poor choice for these kind of planning problems [2]. The A* star heuristics doesn't have this problem since the level sum heuristics is an admissable heuristics, i.e. it never overestimates the cost, and the planning graph heuristics is also consistent, therefore it fullfills the optimality condition [1]. Also the breadth first and uniform cost search are optimal [2].

Table 1: Overview of all results

Algorithm	Air Cargo	Expansions	Goal tests	New nodes	Plan length	Time elapsed
astar_search-h_ignore_preconditions	Problem 1	41	43	170	6	0.0523106
	Problem 2	1450	1452	13303	9	5.33969
	Problem 3	5040	5042	44763	12	20.5908
astar_search-h_pg_levelsum	Problem 1	11	13	50	6	0.673669
	Problem 2	86	88	841	9	58.2366
	Problem 3	365	367	3345	12	387.304
breadth_first_search-	Problem 1	43	56	180	6	0.0424407
	Problem 2	3346	4612	30534	9	15.8045
	Problem 3	14120	17673	123927	12	110.124
depth_first_graph_search-	Problem 1	12	13	48	12	0.014013
	Problem 2	107	108	959	105	0.398715
	Problem 3	3752	3753	30138	293	17.9365
uniform_cost_search-	Problem 1	55	57	224	6	0.0514757
	Problem 2	4853	4855	44041	9	14.3754
	Problem 3	18236	18238	158317	12	61.8446

Listing 1: Plan for Problem 1

```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Listing 2: Plan for Problem 2

```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Listing 3: Plan for Problem 3

```
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

References

[1] Staurt J. Russell and Peter Norvig. "A* search: Minimizing the total estimated solution cost". In: *Artificial Intelligence (A Modern Approach)*. Third Edition (International). Prentice Hall, 2010, pp. 94–99. ISBN: 978-93-325-4351-5.

[2] Staurt J. Russell and Peter Norvig. "Uniformed Search Strategies". In: *Artificial Intelligence (A Modern Approach)*. Third Edition (International). Prentice Hall, 2010, p. 92. ISBN: 978-93-325-4351-5.